GA4-220501095-AA2-EV05 - Desarrollar la arquitectura de software de acuerdo al patrón de diseño seleccionado

Elaborado por:
Andrés Ricardo Bateca Jaimes
Código: 1090384558

ANALISIS Y DESARROLLO DE SOFTWARE

Tutor:
SERGIO IGNACIO MENESES CURDIZ
MELFRY MORENO MOLINA

SERVICIO NACIONAL DE APRENDIZAJE – SENA

COLOMBIA. JUNIO DE 2024

Introducción

En el desarrollo de sistemas de software, es crucial entender cómo se estructuran y despliegan los diferentes componentes que componen la aplicación. Tanto el Diagrama de Componentes como el Diagrama de Despliegue son herramientas poderosas que permiten visualizar y planificar la arquitectura y la implementación de un proyecto de software. Estos diagramas no solo facilitan la comprensión del sistema, sino que también ayudan a coordinar eficientemente el trabajo entre equipos de desarrollo, infraestructura y gestión de proyectos.

El objetivo principal de este trabajo es presentar de manera clara y detallada el diseño de componentes y la arquitectura de despliegue para nuestro proyecto de tienda en línea. Utilizaremos estos diagramas para describir cómo cada módulo y servicio se estructura internamente, cómo interactúan entre sí y cómo se distribuyen físicamente en el entorno de producción.

Evidencia de desempeño: GA4-220501095-AA2-EV05 - Desarrollar la arquitectura de software de acuerdo al patrón de diseño seleccionado

Se entiende por arquitectura de software el conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código fuente.

MVC es un patrón de diseño de software que se divide en tres componentes principales:

- **V**iew (Vista): Se encarga de la presentación y visualización de la información al usuario. Es la interfaz gráfica con la que interactúa el usuario.
- **C**ontroller (Controlador): Actúa como un intermediario entre el modelo y la vista, manejando las solicitudes del usuario, procesando los datos del modelo y actualizando la vista.
- **M**odel (Modelo): Gestiona la lógica de datos del sistema. Es la parte del programa que interactúa directamente con la base de datos.

2. ¿Por qué usamos MVC en el proyecto de la tienda en línea?

1. **Separación de Responsabilidades**

   Modelo: Solo se ocupa de los datos y la lógica de negocio. Esto significa que todas las operaciones de base de datos, como obtener productos, registrar pedidos o gestionar usuarios, se realizan aquí.

   Vista: Solo se encarga de mostrar la información al usuario de manera atractiva y amigable. Es el lugar donde se diseña cómo los datos se presentan al usuario final.

   Controlador: Es el encargado de coordinar las interacciones entre el modelo y la vista. Cuando el usuario realiza una acción, como agregar un producto al carrito, el controlador recibe la solicitud, le dice al modelo que actualice los datos y luego actualiza la vista para reflejar el cambio.

2. **Facilita el Mantenimiento y la Escalabilidad**

   - Separar el código en tres partes distintas permite que cada parte se pueda desarrollar y mantener de manera independiente. Por ejemplo, si se necesita cambiar cómo se muestran los productos, solo se debe modificar la vista, sin tocar el código de la lógica de negocio.
   - Esto hace que el sistema sea más fácil de escalar, ya que se pueden agregar nuevas funcionalidades sin afectar el resto del sistema.

3. **Mejora la Reutilización del Código**

- El uso de MVC permite que el código del modelo (por ejemplo, la lógica para manejar productos) se pueda reutilizar en diferentes partes del sistema sin tener que duplicarlo.
- Además, las vistas se pueden reutilizar con diferentes modelos. Por ejemplo, la misma plantilla para mostrar productos se puede usar para mostrar otros tipos de datos.

4. **Facilita el Trabajo en Equipo**

   Los desarrolladores pueden trabajar en paralelo en diferentes partes del sistema sin interferir con el trabajo de los demás. Un desarrollador puede trabajar en la lógica del modelo, mientras otro se enfoca en la presentación de la vista, y otro en la integración de ambos a través del controlador.

5. **Mejor Experiencia del Usuario**

   Gracias a la separación clara de las capas, es más fácil mejorar la interfaz de usuario y hacer que la experiencia sea más fluida y atractiva. Cambiar el diseño o agregar nuevas funcionalidades a la interfaz no afecta la lógica de negocio, lo que permite iterar rápidamente en la mejora del usuario final.

Diagrama de Componentes Basado en MVC

Para ilustrar cómo se organiza todo en el proyecto de tienda en línea utilizando el patrón MVC, se muestra cómo se relacionan los diferentes componentes.

Componentes del Diagrama

- **Frontend Component**

  Descripción: Es el módulo que interactúa directamente con el usuario final. Se conecta al backend a través de APIs.

  Interfaz: API REST.

- **Backend Component**

  Descripción: Contiene la lógica de negocio y sirve como intermediario entre el frontend y la base de datos.

Subcomponentes:

Controladores: Gestionan las solicitudes del frontend.

Servicios: Contienen la lógica de negocio.

Modelos: Definen las estructuras de datos y se conectan a la base de datos.

Interfaz: API REST, conexiones a la base de datos.

- **Database Component**

  Descripción: MySQL que almacena datos estructurados.

  Conexiones: Conexión directa desde los modelos del backend.

  Interfaz: Conexión SQL.

- **Authentication Service Component**

  Descripción: Maneja la autenticación y autorización de usuarios.

  Interfaz: API REST.

Diagrama de clases del software a construir:

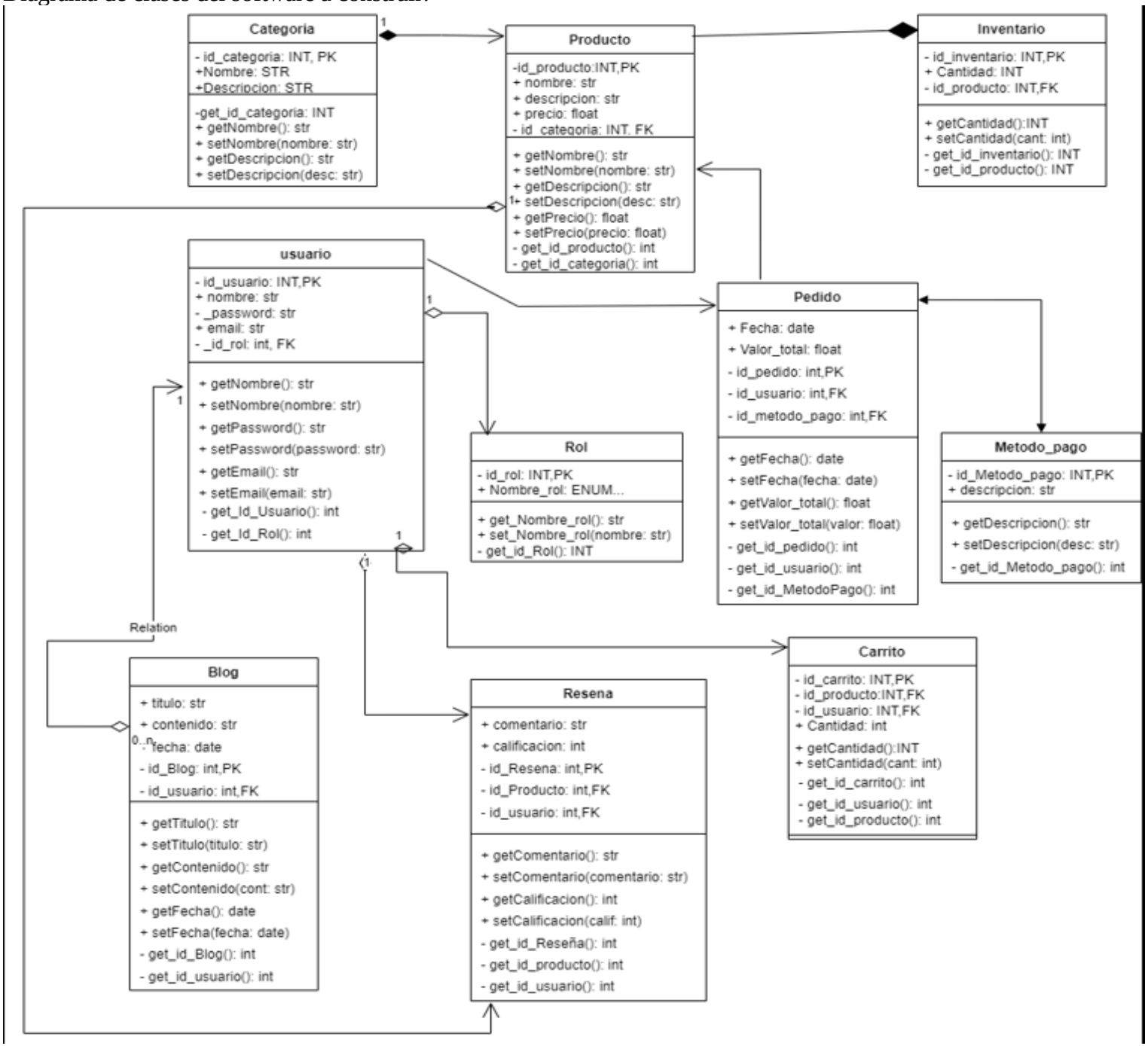
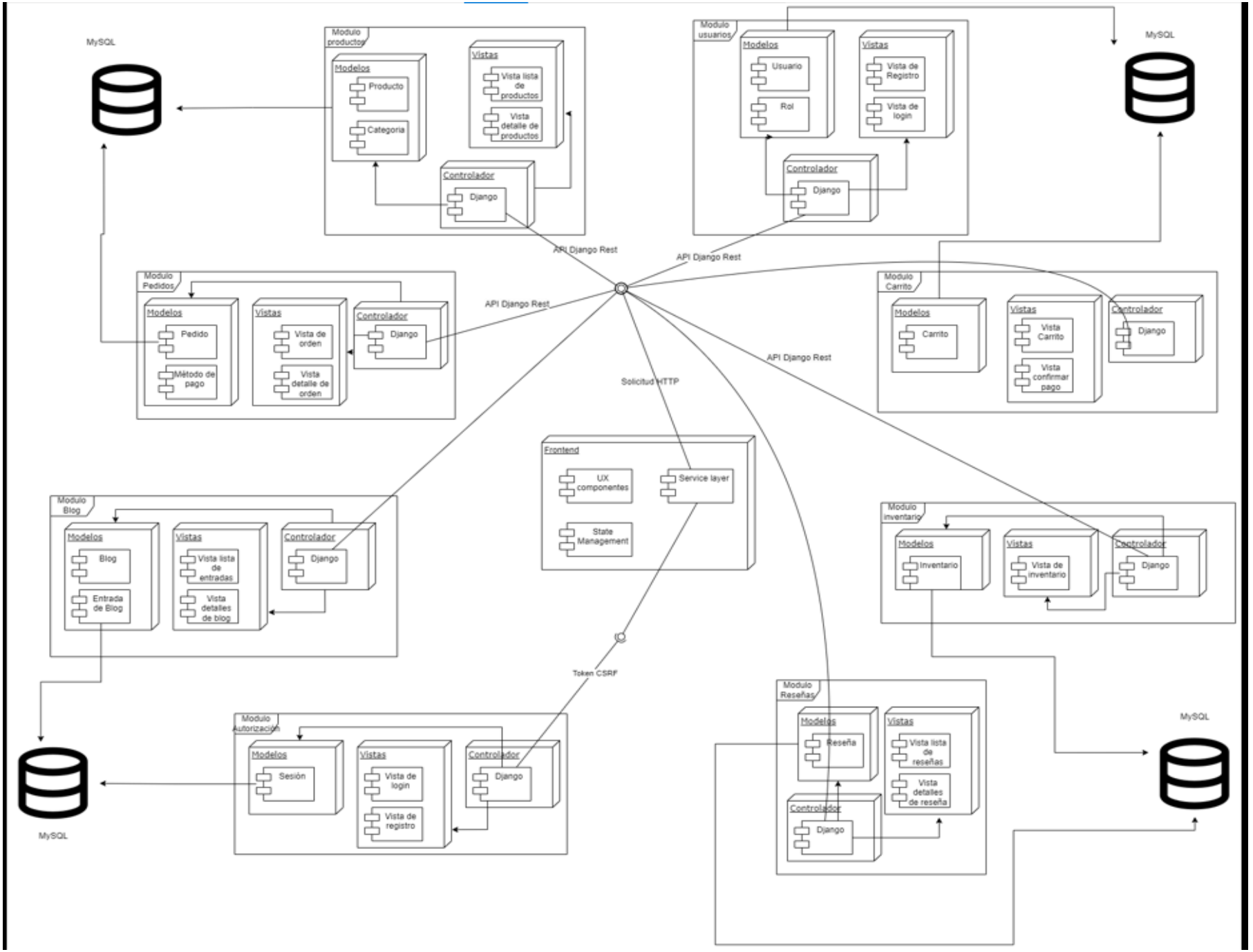


Diagrama de despliegue y de componentes del software a construir:



CONCLUSIONES

El diagrama de clases desarrollado ha logrado representar de manera clara y concisa la estructura fundamental del sistema de software, facilitando la comprensión de las entidades principales y sus interacciones.

Se han aplicado efectivamente buenas prácticas de diseño orientado a objetos en la elaboración del diagrama, asegurando coherencia y modularidad en la estructura del sistema propuesto.

El diseño y la elaboración de los diagramas de componentes y despliegue para el proyecto de tienda en línea han proporcionado una visión clara y detallada de la estructura interna y la distribución física de los componentes del sistema.

La utilización de estos diagramas ha facilitado la coordinación entre los equipos de desarrollo, infraestructura y gestión del proyecto, permitiendo una planificación efectiva y una implementación eficiente de cada módulo y servicio del sistema.

Estos diagramas no solo han servido como herramientas visuales para comprender la arquitectura del software, sino que también han contribuido significativamente a la toma de decisiones sobre la infraestructura de hardware, la escalabilidad y el rendimiento del sistema, preparando el terreno para futuras mejoras y expansiones.